SHARP APL *for MVS*

Internal and Operational Changes

Version 22

SOLITON ASSOCIATES

SHARP APPROACH

Internal and Operational Changes

Version 2.2

SOLITON ASSOCIATES

Internal and Operational Changes

SHARP APL is a registered trademark of Soliton Associates Limited.
All other products referred to in this document are trademarks or registered trademarks of their respective companies.

© 1997 by Soliton Associates Limited — Printed in Canada
44 Victoria Street, Suite 2100, Toronto, Ontario, Canada M5C 1Y2

All rights reserved. No part of this work may be reproduced in any form or by any means without prior written permission of the copyright owner.

Soliton Associates Limited shall not be liable for errors contained herein or for incidental consequential damages in connection with the use of this material. The information contained within this document is subject to change without notice.

Publication code: MM-000-1202 (9703)

Contents

Preface

Introduction vii
Who Should Read This Guide? vii
Conventions viii
The APL Union Character Set viii
Related Publications ix
Contacting Soliton Associates ix

1. Introduction

Overview 1-1
SHARP APL for MVS Products 1-1
Base System 1-1
License Extensions 1-2
DISTNEWS Items 1-2
Messages and Codes 1-2

2. SHARP APL Changes

Overview 2-1
Year 2000 Changes 2-1
SHARP APL Utilities 2-2
Sign-off History 2-2
IFSS 2-3
ATASK Changes 2-3
SVP 2-3
New □*trap* Counter 2-4
WSCACHE Changes - Performance Package Only 2-4
WSCACHE Report 2-4
WSCACHE Function 2-4
New WSCACHE Pagefixing 2-4
WSCACHE Configuration 2-4
File System Storage 2-5

APL Configuration Parameter FILESTOR 2-5
APL Modify Command FILESTOR 2-5
Privileged APL Function *filestor* 2-6
Storage Allocation Changes 2-6
Residue Function 2-6
NSVP Remote User Names 2-6
SNAP Dump Support 2-6
Sign-on hangs 2-7
APLUADxx Parmlib Member 2-7

3. *The Shared Variable Processor*
Overview 3-1
SVP 2.2 Changes 3-1
SVP Reliability 3-1
Version Checking 3-1
Autoloading the SVC Code 3-2
SVP Resource Management 3-2
SVP Monitor 3-3

4. *File System Changes*
Overview 4-1
Track Buffering Control 4-1
Redesigned File System Histograms 4-2
Improved File System Startup 4-2
Online Condense 4-2
Rename of the READ Relo Module 4-2

5. *Changes to SHARP APL Utilities*
Overview 5-1
Improvements To File Restore Operation 5-1
Online Backups 5-1
BSUT File System Utility 5-2
BSCR File System Crash Recovery 5-2

6. *Changes to APL-Coded Software*
Overview 6-1
Year 2000 Compliance 6-1
Changes to Files: 6-2
Changes to Functions and Variables 6-2
B-Task Scheduler Shutdown: Delay Parameter Added 6-6
Changes to Operator Functions 6-6
crashfile (workspace 314158 *state/opr*) modified 6-6

spstatus (workspace 314158 *opr/opfns*) modified 6-6
fadd (workspace 314158 *opr/opfns*) modified 6-7
finq added (workspace 314158 *opr/opfns*) 6-7
lately added (workspace 314158 *opr/opfns*) 6-7
filestor added (workspace 314158 *state/opr*) 6-7
trapcnt added (workspace 314158 *state*) 6-7
trkhit (workspace 314158 *opr/opfns*) modified 6-7
Performance Package Functions - Documentation Added 6-8
Changes to 1854339 *dump* Workspace 6-8
bsshist function added 6-8
wssi function modified 6-8
New Debugging Tool 6-8

7. Environment Issues
IBM APAR OW22019 7-1
IBM APAR OW16355 7-2
Configuring SRM for SHARP APL 7-2

8. APLMON
Overview 8-1
New and Changed Reports 8-1
New Report: *bsstb* - the Track Buffer Report 8-1
Changes to the *bss* Report 8-1
Changes to the *wscache* Report 8-1
Changes to the *except* Report 8-2

Preface

Introduction

SHARP APL for MVS Version 22 (V22) provides a number of enhancements to the SHARP APL product, providing improved performance, increased availability, and enhanced operability of the SHARP APL system. These changes include modifications and additions to the base APL system, as well as changes to the SHARP APL utilities and other related software products.

The *SHARP APL for MVS Internal and Operational Changes* guide describes the changes to SHARP APL since V21. It includes changes to SHARP APL which are transparent to users, but which might be apparent in the performance perceived by those users or by other systems that share the computing facilities.

This document, when used with the *SHARP APL for MVS Guide for APL Programmers*, completes a summary of all changes to SHARP APL and related products distributed as V22. Detailed procedures for upgrading to V22 are provided in the *SHARP APL for MVS Upgrade Guide*.

Note: Rather than reproduce existing material, references to related SHARP APL publications are supplied where applicable.

Who Should Read This Guide?

This guide should be read by personnel responsible for the operation and maintenance of SHARP APL for MVS.

Conventions

The following conventions are used throughout this guide.

`⎕io←0`	All examples in this guide assume that index origin is set to `0`.
`abcde...` *`ABCDE....`*	Examples of input to and output from SHARP APL are shown in the APL union character set, in which the first and second APL alphabets appear in lowercase and uppercase, respectively. For more information, see "The APL Character Set" below.
`constant width`	Examples of non-APL input and output (e.g., JCL) are represented in a `constant width` typeface.
passno	Argument variables appear in *sans-serif italic* type.
`[ ]`	Optional arguments are shown between square brackets. For example, in the following APL statement, the argument *passno* is optional:

'filename' `⎕stie` *tieno* `[`*,passno*`]`

The APL Union Character Set

The names of APL files and workspaces are constructed from the characters *a–z*, *A–Z*, `0–9`, and the glyphs `Δ` and `⍙`. The characters *a–z* are referred to as the first alphabet, and appear as lowercase letters in this guide. The characters *A–Z* are referred to as the second alphabet, and are represented as uppercase letters.

Due to the wide variety of input and output devices available for SHARP APL, these alphabets may display differently on your terminal. For example, the first alphabet is often displayed as uppercase, while the second alphabet is displayed as underscored uppercase. To determine what the first and second alphabets look like on your terminal, display the locations `⎕av[86+⍳26]` and `⎕av[113+⍳26]`.

APL workspace and file utility selection statements represent these characters as follows:

In APL Union	In EBCDIC	In Older APL Systems
a–z	A–Z	A–Z
A–Z	a–z	A–Z
0–9	0–9	0–9
Δ	- (minus)	Δ
⍙	+ (plus)	⍙

The display of names is controlled by the `EXEC PARM` parameter `LOWERCASE=YES | NO`. If `LOWERCASE=YES` is specified, then the above convention is also used to write to `SYSPRINT`. If `LOWERCASE=NO` is specified, then second alphabet APL characters are displayed in uppercase, and a second line is printed, without carriage advance, underscoring these letters.

Related Publications

The following manuals are distributed with SHARP APL for MVS Version 22. The system administrator should ensure that these manuals are available.

SHARP APL for MVS,

- *Internal and Operational Changes,* publication code MM-000-1202
- *Messages and Codes,* publication code MM-000-1602
- *Guide for APL Programmers,* publication code MM-000-1502
- *System Administrator's Guide,* publication code MM-000-1002
- *System Maintenance Guide,* publication code MM-000-1102
- *Upgrade Guide,* publication code MM-000-0202
- *Utilities Manual,* publication code MM-000-1302
- *SVP Configuration, Operation, and Maintenance Guide,* publication code MM-001-0301
- *Interface to MVS JES Configuration, Operation, and Maintenance Guide,* publication code MM-000-1302

The following manuals represent the SHARP APL for MVS product extensions that were updated with Version 22:

SHARP APL for MVS,

- *Usage Inquiry System User's Guide,* publication code MM-031-0401
- *BTASK Scheduler Configuration, Operation, and Maintenance Guide,* publication code MM-031-0301
- *APLMON User's Guide,* publication code MM-006-0402

For a complete list of SHARP APL for MVS publications, please visit the Soliton Associates website at `http://www.soliton.com/services.html`.

Contacting Soliton Associates

Problems or questions regarding your SHARP APL system or this guide should be directed to the Soliton Associates Technical Support group, which can be reached using either of the following methods:

- SHARP APL MAILBOX to zdist
- Internet mail to `zdist@soliton.com`

Requests for additional technical documentation should be directed to your account manager or the Soliton Associates Sales group, which can be reached using either of the following methods:

- SHARP APL MAILBOX to sales
- Internet mail to `sales@soliton.com`

1 Introduction

Overview

SHARP APL for MVS Version 22 (V22) is an upgrade to SHARP APL for MVS Version 21. Changes to SHARP APL in V22 include modifications and additions to the base APL system, as well as some changes to the SHARP APL utilities and other related software products.

SHARP APL for MVS Products

The following products are distributed with V22:

Base System

APL Supervisor	(ASUP)
APL File Crash Recovery Utility	(BSCR)
File Processor	(BSON)
File System Tape Copy Utility	(BSTC)
APL File System Utilities	(BSUT)
File System Online Utility	(FUTL)
328X Printer Support	(HCPR)
User Number Maintenance Facility	(NREQ)
APL Language Processor	(SAPL)
Shared Variable Processor	(SSVP)
APL Workspace Backup Utility	(WSUT)
Workspace Tape Copy Utility	(WSTC)

License Extensions

Some products are distributed as individual upgrades or as completely new releases. The following products can be licensed separately as extensions to your base system.

Billing/Usage System	(BILL)
Batch APL Support	(BTSK)
MVS JES Interface	(IFSS)
APL System Performance Monitor	(MON)
Performance Package	(PERFP)
Queue Management Server	(QMS)
Resource management	(RADMS/FS)
APL File Conversion Utilities	(SHOV)

DISTNEWS Items

General maintenance and improvements to SHARP APL for MVS since Version 21 provide a system that is more reliable, more efficient, and easier to use. These changes include DISTNEWS items 659, 857, 858, 863, 865, 866, 867, 869, 870, 871, 872, 877, 880, 881, 883, 884, 885, 887, 888, 890, 891, 892, 893, 894, and 895.

For information on these and other items, refer to the documentation in the `707 distnews` workspace on the Soliton Lab system.

Messages and Codes

The following messages issued by APL and its utilities are either new for this release, or have been modified since V21. See the *SHARP APL for MVS Messages and Codes Manual* for details.

APL0052	CHANGED
APL0177	CHANGED
APL0189	NEW
APL0235	NEW
APL0300	NEW
APL0386	NEW
APL0400	NEW
APL0401	NEW

APL0402	NEW
APL0446	NEW
APL0447	NEW
APL0471	CHANGED
APL0472	CHANGED
APL0530	CHANGED
APL0661	CHANGED
APL0701	NEW

2
SHARP APL Changes

Overview

The changes made to the SHARP APL for MVS for V22 are outlined below. They are discussed in more detail in the sections that follow this overview.

- Year 2000 changes
- IFSS has been introduced as a new mechanism for transporting JES data to APL.
- ATASK changes have been made to permit new diagnostic tools, including a new □*trap* counter.
- WSCACHE now provides control over the percentage of free space.
- APLCNFxx-PARMLIB Configuration Change
- Storage allocation changes have reduced APL's dependence on real (physical) memory.

Year 2000 Changes

Soliton Associates have used good faith efforts to review and, where required, change SHARP APL for MVS to be Year 2000 compliant.

There have been several changes to the structure of data sets and files used by the billing system, and to the SHARP APL utilities to allow for the appropriate storage and representation of dates so that date calculations and date information are accurate across the century change. Where two digit years continue to exist, if the year is in the range 50 to 99, the system assumes years 1950 to 1999; if the year is in the range 0 to 49, it assumes 2000 to 2049. Changes to the various system components will enforce four digit years in future releases, where possible.

SHARP APL Utilities

The SHARP APL utilities support the option `RETPD=` and also let the user specify `RETPD` (or `EXPDT`) via JCL. The IBM solution is to declare that the byte that holds *yy* of *yyddd* is an offset from the year 1900. This is a three byte field: byte 0 is *yy*, bytes 1-2 are *ddd*. This solution will work until 2155 (1900+255). This is the only data-sensitive area in the SHARP APL utilities.

Sign-off History

The sign-off history data sets and files have been changed to allow for the millennium date change. These changes affect the sign-off history data set (`SAPL.DIST.SOH`), the unformatted sign-off history file (`314159` *ohc*), and the formatted sign-off history file (`314159` *ohcf*).

Before V22:

- The actual SOH record contains (beginning at byte 8) the sign-off date encoded in eight bytes as *mm/dd/yy* (in qglyphs).
- The file system copy of this record (`314159` *ohc*) contains the same thing.
- The formatted version (`314159` *ohcf*) contains, in column `1+⎕io`, the date in a four byte integer of the form: *mmddyy* (and the time of day in word 2).

After V22 Upgrade:

- In the SOH record, the time of day clock is now stored in bytes 8-15 encoded as *yyyymmdd*.
- In the SOH record, bytes 16-19, which formerly contained sign off time, are now zero-filled and reserved for future use.
- The above two changes are reflected in both the sign-off history data set and the unformatted file.
- The formatted file contains the time of day clock in words 1 and 2.
- The directory at the beginning of the formatted file, which contains the date as *ymmddhhmm*, is unchanged for historical data. New directory entries following the upgrade are stored as *yyyymmddhhmm*.

Implications:

- SHARP APL products which reference the sign-off history records, BILL and NREQ, have been revised to work with these changes.
- If there is any private code on your system which references any of these files, or the data set, it should be revised to reflect these changes.
- As soon as the V22 system is started, sign-off history records are written in the new format. V21 system software will not be able to interpret some of the new date formats. At this point, it will be very difficult to regress to V21 without the loss of SOH records. Soliton recommends that you immediately

run tests which will allow you to commit to the new version before any production work is started. Additional details are available in the *Upgrade Guide* for V22. Contact *zdist* if you find it necessary to regress to V21.

Information on the other changes made for Year 2000 are in Chapter 6 of this guide.

IFSS

IFSS is a new mechanism for transporting JES data (SYSOUT) into APL. This new SHARP APL started task implements a functional sub-system interface to JES which supports definition of multiple SYSOUT print queues. The existing IMVS product is unchanged for customers who prefer not to convert. The IBM interface for the IMVS product is relatively stable and no improvements are scheduled. The new mechanism is essentially the same as IBM's support for their advanced function printers.

A Functional Sub-System (FSS) is an address space started by JES which performs processing for JES, and remains loosely controlled by JES. A Functional Sub-System Application (FSA) is a task running in an FSS address space which is associated with a particular device within JES. IFSS is a standard FSS and, within it, one or more FSAs will normally be active.

Migration to IFSS is optional. The conversion process requires careful consideration of SHARP APL and MVS environment issues. Even though IFSS is distributed with V22, Soliton recommends customers complete the V22 upgrade before considering conversion to IFSS. Customers who would most benefit from the conversion are those who experience performance problems due to the serial processing of IMVS. In IFSS, on the other hand, if a job generates three data set groups and there are three IFSS printers (FSAs) matching the data set groups' characteristics, then each FSA will be assigned one of the groups, all at the same time.

For more information, consult *SHARP APL for MVS JES Interfaces Configuration, Operation and Maintenance Guide.*

ATASK Changes

Several changes have been made to ATASKS that add diagnostic capabilities.

SVP

The ATASK has been lengthened to include an SVP posting history vector. A new trace type has been added, and all SVP posts into APL produce these trace entries if 'SVP tracing' is enabled.

New □trap Counter

An ATASK cell has been added to count how many times a task executes □*trap*.

The trap counter is set to 0 at signon, and increases by 1 every time an error is trapped, up to a theoretical maximum of x'7fffffff'. The trap count cell in the ATASK is located at offset x'104', or 260. Every task's trap count is reduced by 50% every five minutes, and the count is cleared upon return to immediate execution.

A new exception report is available in APLMON 3.2 which can identify offending tasks.

WSCACHE Changes - Performance Package Only

WSCACHE Report

The WSCACHE address and count of forced swaps to disk from WSCACHE have been added to the WSCACHE report, and to message APL0052.

WSCACHE Function

The old *wscache* function has been removed from `314158 opr`. This function, distributed with Version 21, is now obsolete. If you have used this in any private code, it should be removed.

A new *wscache* function has been added to `314158 state`. This function allows the same manipulation of the WSCACHE that is permitted by the MVS command "MODIFY". The function accepts the suffixes '*m*' and '*M*' for WSCACHE size, and interprets them as ×1024×1024.

New WSCACHE Pagefixing

When the WSCACHE is changed on a running system, pagefix is done in increments of 100k or less per trip through the scheduler's periodic scan so as to reduce the impact on MVS.

WSCACHE Configuration

The WSCache configuration has been extended to allow the administrator to define the amount of free-space. In Version 21 this value was fixed at 25%. A new subparameter, `pctfree`, has been added to the APL configuration. Its syntax is shown in the example:

```
wscache=(5000k,3000k,nofix,pctfree=25)
```

The value for the new `pctfree` subparameter must be in the range [0,100]. The subparameter specifies the percentage of the usable cache (3000k in the above example) that the scheduler will try to keep free, using otherwise unused swap

bandwidth, in periods when APL's workset size exceeds the usable size of the cache. The administrator is not required to specify the `pctfree=` sub-parameter in the APL configuration; the default value of 25% is used if no value is specified. There has also been an addition to the `wscache` modify command:

```
f apl,wscache 3000k,nofix,pctfree=25
```

File System Storage

APL Configuration Parameter FILESTOR

The syntax of the FILESTOR=APL configuration parameter has been extended to:

FILESTOR=(FIX | NOFIX,FIX | NOFIX)

The first sub-parameter controls file system storage below 16m. The second sub-parameter controls file system storage above 16m.

FIX	pagefix the storage.
NOFIX	do not pagefix the storage.

The default is FILESTOR=(FIX,FIX). If only one sub-parameter is supplied (FILESTOR=FIX | NOFIX), it applies only to storage below 16m, and storage above 16m is pagefixed (the default). If you do not change the FILESTOR= parameters from their current settings, you will get exactly the same behavior as with the previous release.

APL Modify Command FILESTOR

The FILESTOR sub-command has also been added to the MVS MODIFY command for APL. This allows file system storage to be adjusted as needed.

F APL,FILESTOR FIX | NOFIX,FIX | NOFIX

The first parameter controls file system storage below 16m. The second parameter controls file system storage above 16m.

FIX	pagefix the storage if it is not already pagefixed.
NOFIX	pagefree the storage if it is not already pagefree.

The result of the command is the display of message APL0446, which reports the current settings (after implementing the command). If no parameters are supplied (F APL,FILESTOR), the only action taken is to display message APL0446 reporting the current settings.

All storage that is pagefixed with this command is allocated in 100k increments, with one pagefix performed per trip through the APL scheduler's periodic scan.

Privileged APL Function `filestor`

In `314158 state`, there is a new privileged function called `filestor`. It provides the functionality of the MODIFY sub-command FILESTOR. The syntax is:

```
filestor 'fix/nofix,fix/nofix'
```

The first parameter controls file system storage below 16M. The second parameter controls file system storage above 16M.

fix	Pagefixes the storage if it is not already pagefixed.
nofix	Pagefree the storage if it is not already pagefree.

Result of the function is the text of message APL0446, and message APL0446 is also displayed on the system console. If an empty argument is supplied (`filestor ''`), the result is returned and message APL0446 is displayed on the system console, but no changes are made to the status of file system storage.

All storage is pagefixed in 100k increments, one pagefix per trip through the APL scheduler's periodic scan.

Storage Allocation Changes

APL's dependence on real (physical) memory has been reduced. Most memory used by APL, the APL file system, and the utilities now use backing mode 'ANY'. This should contribute to lowered utilization of 'BELOW' memory, and eliminate unnecessary page migration during EXCP-intensive operations.

Residue Function

Prior to this release, when `⎕ct` was set to zero, floating point arguments to the residue function (`|`) caused abnormal results or system errors with a near-zero PSW address. This problem is fixed with V22.

NSVP Remote User Names

The APL limit on the NSVP remote user name is extended to 20 characters from 8 characters. This allows shares with names of the form *name@host*.

SNAP Dump Support

Reference to the obsolete 24-bit "`insert storage key`" instructions (`ISK`) has been removed from the snap dump support module `ISNP`. Some versions of the assembler now treat `ISK` as an invalid op code. This instruction has been

superseded by the "`insert storage key extended`" instruction. While the code containing the ISK instruction was not being used, its presence in the module caused assembly errors on some systems.

Sign-on hangs

In Version 20 Performance Package and Version 21 base system, improvements were made to the STASK code to remove extra posts. Since then, on systems which have a low STASK activity and also have a bad offer outstanding to AP-1, attempts to sign on can hang after typing the sign-on text. The offer was ignored until STSK was posted again for some other reason. With Version 22 the returns from SCAN are improved so users should not experience this problem.

APLUADxx Parmlib Member

The APLUADxx member is only used when the system uses an early version of IBM's VSAM. On the majority of systems this member is no longer required. The APL procedure has been updated to remove steps related to crash recovery of the UADS data set.

3 The Shared Variable Processor

Overview

The changes made to the Shared Variable Processor for V22 are outlined below. They are discussed in more detail in the sections that follow this overview.

- The reliability of the Shared Variable Processor has been improved.
- Version checking has been added for all components of the SVP subsystem.
- The SVP may now be used to install SVC code when it starts.
- The SVPs resource management has been improved.
- The SVP now has an interval-driven monitor.

SVP 2.2 Changes

SVP 2.2 includes packaging and reliability improvements; there are no user-task visible changes included in this release of the SVP.

SVP Reliability

Improvements to the Shared Variable Processor (SVP) address a class of problems which may occur when an MVS task using the SVP experiences an unexpected address space termination. The SVP processing of such events has been reworked, resulting in an improved ability to continue execution without degradation.

Version Checking

All the components of the 2.2 SVP subsystem are version-stamped, and an internal interface-level scheme has been introduced to prevent incompatible components from interacting. The SVC module has had this information included in EBCDIC so that a browser or other display utility may be used to verify its release level.

SVP 2.2 sub-components (SVP and SVC) will refuse to interact with earlier SVP releases; these checks did not exist in the earlier releases, so care must be taken that SVPs at level 2.2 are not used with earlier SVP release components. A failure of this sort could occur while testing the V22 upgrade. Accidentally, a pre-2.2 SVC or APL system could be used to access an SVP 2.2. Usually an SOC1 abend will occur, but occasionally the failure mode will not be obvious.

Autoloading the SVC Code

The SVP may be used to install SVC code when it starts. This is an option which, if used, can improve the ease of installation of the SVP. This would eliminate the need for IPLs during the upgrade procedure.

The SVC code shipped with V22 may be installed in the normal manner, if desired. The *V22 Upgrade Guide* also provides instructions so that, once the new SVP is in production use, the new SVP place-holder module YSVS is installed. The sole purpose of the SVC stub is to be a place holder for a dynamically loaded SVP SVC. Most customers will find this the best alternative; once the SVC stub is installed, no further IPLs will be required for subsequent SVP upgrades.

The only case where the dynamically loaded code is not the best solution is when a customer needs to use the same SVP SVC to access more than one SVP. In this situation, the YSVC SVC code must still be statically loaded and will need to be upgraded with most upgrades of the SVP.

The SVC code that is dynamically loaded is the same as the current statically loaded SVC, except that dynamically loaded SVCs use a stub area in the SVP (referred to as a "dynamic SVC slot") which includes a dynamically built copy of the YCOM area. This means that dynamically loaded SVCs do not use the SVC number or SVC subsystem name provided by SITELCL, even though the code does have these values assembled into it locally. It also means the same care must be exercised in configuring a dynamic SVC with the correct SVC subsystem name as that required to compile a static SVC via the SITELCL mechanism.

SVP Resource Management

SVP 2.2 no longer uses the subsystem end-of-task (EOT) hook. This hook meant that each SVP received control as each task in the system was ending. This was not a performance problem, as the amount of processing for a non-SVP task was small, but it was, on occasion, an integrity problem. SVP 2.2 uses the RESMGR mechanism to add resource managers for tasks and address spaces that are actually using the SVP.

SVP 2.2 uses the subsystem end of memory (EOM) hook for a quick check when an address space ends; no more than a few instructions are required if the address space either did not use the SVP or has already been cleaned up by the resource manager code.

The emergency cleanup code triggered by the EOM checks is run from the new monitor in the SVP address space.

SVP Monitor

The SVP now has an interval-driven monitor. This code is meant to react automatically to abnormal situations rather than waiting for an operator to notice them.

In this release, the monitor has only one function; emergency EOM cleanup. The subsystem EOM hook merely sets a bit when it detects that an address space has not cleaned up properly when it terminates; the SVP monitor periodically checks these bits and it will run a proper address space cleanup if it sees that one is required.

4 File System Changes

Overview

The changes made to the File System for V22 are outlined below. They are discussed in more detail in the sections that follow this overview.

- The V22 file system includes several enhancements to track buffering:
- File System Histograms have been redesigned to eliminate memory consumption below the line and to provide new histograms.
- A phase identity test has been added to the file system startup.
- An improvement has been made to the online condense process.
- The spelling of a module name has been changed to avoid a conflict.

Track Buffering Control

The V22 file system includes several enhancements to track buffering:

- The logic for searching track buffers has been enhanced to employ a binary search technique. This is intended to reduce CPU usage where very large arrays of track buffers are configured.
- A new file system configuration parameter: `TRACKLEN=` which provides an alternate way to specify track buffer length. Previously this was specified by `PHRPT=`, but this calculation could produce very large buffers.
- A new file system configuration subparameter to the VC= parameter called `TRACK=` allows track buffers to be configured at the volume class level. Possible settings are:

`TRACK=NO`	gives no track buffering within this Volume Class
`TRACK=YES`	gives permanent track buffering within this Volume Class
`TRACK=ADAPTIVE`	gives track buffering as for as long as it enhances performance

Redesigned File System Histograms

In V22 histograms are kept above the line.

The file system histograms have been redesigned to provide higher resolution for the information recorded. No external configuration information is required in order to use the new histograms. New file system histograms included in Version 22 will:

- measure CPU time per file operation,
- measure elapsed time to perform workspace swaps during file operations,
- measure track buffer performance

Using these histograms, experienced system programmers can tune track buffering to suit their own file usage patterns. A new function (*bsshist*) has been added to the *dump* workspace.

The FILESIZE= parameter in the APL configuration may need to be increased to accommodate the new histograms. An additional 64K is required in above the line storage.

Improved File System Startup

The file system startup now includes a test to verify that the phase starting actually is the file system. If the phase is not BSON, an error message is issued by the *startfiles* function indicating an incorrect link.

Online Condense

Before this release, if a file's directory information was found during file condense, but the file was not tied, the condense operation would update the TIFI without interlocking *perfile*. This can permit another operation that uses perfile to run concurrently with the condense, leading to a file system program check when the resulting buffer conflict is detected.

Because this file program check was encountered during *perfile* manipulation, it could cause subsequent *perfile* related file operations (such as □*tie*, □*untie*, □*create*, and □*erase*) to hang waiting for the crippled serial processor to release *perfile*.

Rename of the READ Relo Module

The spelling of the module READ has been changed to REED in order to avoid a conflict with an IBM macro of the same name.

5 Changes to SHARP APL Utilities

Overview

This chapter lists changes involving SHARP APL utilities. For detailed information on the function and use of SHARP APL utilities, refer to the SHARP APL *Utilities Manual*.

Improvements To File Restore Operation

The support for restore placing file or libraries on specific extents has been improved. Only the files and libraries specified for the extent are placed on the extent, unless there is absolutely no space on other extents.

If the restore to an extent fails due to insufficient disk space, message APL0447 is issued. Previously this failure caused an abnormal termination.

A new option, used in Restore, has the syntax:

`INCLUDEONLY=YES/NO` [default is `NO`]

Prior to the introduction of the option (or when the default value is used), all files are restored except those excluded by 'exclude' cards. 'Include' cards are permitted, and can be used to specify an extent, but they are not required.

If `INCLUDEONLY=YES` is specified, only system files (*`perfile`*, *`permit`*, *`perhash`*) are restored, plus any files, libraries, or owners specified by include cards. Exclude cards are ignored.

Online Backups

In previous releases a race condition between file system interlock and the erase and rename operations could cause a file system program check during online backups. This has been resolved in V22.

BSUT File System Utility

The restriction limiting file access matrices to 1000 rows has been removed. The BSUT utility uses a pseudo-workspace for processing access matrices; in the case of very large matrices, this workspace can overflow, resulting in a message indicating first the current workspace size, and second the calculated size needed for processing. The new option `WSSIZE=` is used to resize the pseudo-workspace.

The dump utility now tests to see if disks are partially restored. If so, the dump cannot be performed, and message APL0177 is issued.

BSCR File System Crash Recovery

New behavior as of MVS 5.1 can cause the file system crash recovery procedure to fail with a JCL error. The system ignores a number of keywords when they are specified with the START command, unless they are coded with no values to nullify the same keywords in the source JCL. This list of keywords includes CLASS= which is used by the sample BSCR procedure. DD or JOB statement keywords cannot be used as JCL symbols in procedures or jobs that are started by a START command from the MVS console.

The keyword CLASS= is changed in the BSCR procedure CPYBSCR to VCCLASS=. Before upgrading to MVS 5.1, the locally configured BSCR procedure on your system should be changed in the same way.

6

Changes to APL-Coded Software

Overview

This chapter discusses changes involving APL-coded software such as those required to deal with the year 2000, changes to the B-task scheduler, the operator environment, resource administration facilities, and the dump analysis system.

Year 2000 Compliance

The APL-coded software included in Version 22 has been updated to allow storage and representation of dates necessary to correctly deal with the century change. The changes include modification to the internal representation of the Usage data file (`314159` *`ohc`*) and the formatted version of the file (`314158` *`ohcf`*), as well as changes to a number of APL functions used in the Usage Inquiry system and elsewhere.

NOTE

Authors and users of private copies of APL-coded software supplied by Soliton are urged to read this section carefully, and consider the implications; because the underlying structure of the sign-off history files (offhist) has changed, and because many reporting functions in the Usage Inquiry System now behave differently, private copies may fail or provide incorrect results.

Changes to Files:

- 314159 *ohcmmyy* (the unformatted offhist file) is named 314159 *ohcmmyr* where *yr*=1900|*yyyy*.
- 314159 *ohcfmmyy* (the formatted offhist file) is now named 314159 *ohcfmmyr* where *yr*=1900|*yyyy*. **New format directory entries** are stored as *yyyymmddhhmm*, while **old format directory entries** remain *ymmddhhmm*. **New format data entries** store the date as *yyyymmdd*, while **old format data entries** remain *mmddyy*.
- 444 *bfmmmyy* (the monthly consolidated usage/billing file) is now named 444 *bfmmmyr* where *yr*=1900|*yyyy*. All status dates in component 1 are *yyyymmdd*.
- 1708363 *bfind* (tracks daily and monthly billing runs) now stores the component 2 date (date of the last daily billing run) as *yrmmdd* where *yr*=1900|*yyyy*.
- 444 *numbreq* (*numbreq* data file) holds dates in component 17 as *yyyymmdd*. Directory components [;6+⎕*io*] for **new format directory entries** are *yyyymmdd*, while **old format directory entries** remain *yymmdd* and are assumed to be for years 19*xx*.

Summary of Billing File Name Changes

The files 314159 *ohc*, *ohcf* and *bfmmm* form the year portion of their names from the pattern: 1900|*yyyy*, resulting in file names like *ohcf*100 and *bfmmm*100 for the year 2000. For example:

Year	File Names	Year	File Names	Year	File Names
1997	*ohcmm*97 *ohcfmm*97 *bfmmm*97	2000	*ohcmm*100 *ohcfmm*100 *bfmmm*100	2001	*ohcmm*101 *ohcfmm*101 *bfmmm*101

Changes to Functions and Variables

All software was modified, where appropriate, to recognize four-digit years, while compatibility was maintained with the two-digit format formerly used. The following table shows a summary of the changed functions and variables grouped by the workspace or file that contains them.

Note: Please see the end of this multi-page table for notes.

Workspace/File	*Function/Variable Name*	*Change*
314159 *use*	*Δnumbdelete*	The delete date is stored as *yyyymmdd*
	Δnumbxfer	The transfer date is stored as *yyyymmdd*
	Δohformdir	Make offhist directory entry *yyyymmddhhmm*.
	Δohpend	Modified to handle offhist naming convention.
1708363 *accnt*	*accnt*	Fileprint accounting turned off. *Bfind* component 2=*yrmmdd*, where *yr*=1900|*yyyy*.
	bfadd	*RA=mm,yy* or *mm,yyyy*. If 2 digit year use the 0-49, 50-99 rule.†
	bfpurge	"
	bftie	Handles new *bfmmmyr* name convention.
	billhsp	Status date stored as *yyyymmdd*.
	billsort	"
	sohact	"
	storact	"
	ohtie	Handles new offhist file naming convention
1854339 *userlog*	*convtstamp*	Formats date as *yyyy,mm,dd*.
	nextday	Handles 2 or 4 digit years.
	nextmonth	"
	printdates	Formats year to 4 digits.
	to	*CDATE* year is now 1900|*yyyy* not 100|*yyyy*.
	prevday	Handles 2 or 4 digit years.

†

Workspace/File	*Function/Variable Name*	*Change*
567 *violsys*(f)	*findplace*	Data lookup uses 1900|*yyyy* for year.
	getdata	Change to date lookup. Now treats year as 1900|*yyyy* not 100|*yyyy*.
	violog	Formats date as 4 digits: *yyyy*.
	chronrep	Prompts for 4 digit year.
	dailyrep	"
	dumprep	"
	fullrep	"
	mergerep	"
	mergesub1	Date breakdown preserves 4 digit year.
	missrep	Prompt for 4 digit year.
	monthlyrep	"
	nlrep	"
	sessionrep	"
	userrep	"
	trans	Preserve 4 digit year.
	tiedata	Handles offhist file naming convention
	getdata	Handles both new and old offhist date formats.
	findplace	Handles both new and old offhist directory formats.
314158 *numbreq*	*activity*	Date prompt is for 4 digit year.
	limits	Ensures transfer/delete dates are of *yyyymmdd* format.
	nreq	Uses full □*ts*, not 1900|.
	offhist	*LA* can be either *mmyy* or *yyyymm*. If *mmyy* use the 0-49, 50-99 rule to determine the year. Also takes into account the new offhist file naming convention. †
	Δ*log*	Logs request date as yyyymmdd not yymmdd.
	Δ*tad*	Format year to 4 digits.
	FMT(*v*)	Year formatted to 4 digits.
1 *usagesystem*(f)	*desc1*(*v*)	Updated to reflect 1 *usage* changes. (*desc1* is the describe for the workspace)

Workspace/File	*Function/Variable Name*	*Change*
1 *usage*	*ohprint*	*RA* can be *mm,yy* or *mm,yyyy*. If year is 2 digits then the 0-49 etc. rule is applied.†
	print	"
	time	Handles formatting of *mmddyy* and *yyyymmdd* dates.
444 *bfmaster*(f)	*offhist**	Dates returned are *yyyymmdd* not *mmddyy*. *RA* can be *mm,yy* or *mm,yyyy*. If 2 digit year the 0-49 etc. rule applies.†
	*ohonline**	Handles new *OHCF* naming convention.
	*storage**	*RA* can be either *mm,yy* or *mm,yyyy*. If year is 2 digits then the 0-49 rule applies.†
	*updates**	Handles new *bfmmmyr* file naming convention. *RA* can be *mm,yy* or *mm,yyyy*. If year is 2 digits the 0-49 etc. rule applies.†
	*usage**	*RA* can be *mm,yy* or *mm,yyyy*. Again, the 0-49 etc. rule applies for 2 digit years.†
1854339 *storage*	*storage*	Prompts for date as *mmmyyyy*.
	storagebyuser	"
314158 *state*	*crashlog*	Ties offhist file using new naming convention.
	crashfile	"
1 *ts*	*ttf*	*RA* can have 2 or 4 digit year. If 2 digit year then 1900 assumed.
	lts	"
	utc	"
	getoffset	*LA* can have a 2 or 4 digit year. If year is 2 digits then 1900 is assumed.
	ttfhow(v)	modified to reflect change to function.
	ltshow(v)	"
	utchow(v)	"
	getoffsethow(v)	"

Notes:
** =this is called by a cover function in workspace* 1 *usage.*
RA=right argument.
LA=left argument.
(f)=file.
(v)=variable.
† The 0-49, 50-99 Rule: if the year is in the range 50-99, it assumes years 1950 to 1999, if in the range 0-49, it assumes 2000 to 2049. (If yyyy is used, then yyyy is a simple year value, such as 1997 or 2000.)

B-Task Scheduler Shutdown: Delay Parameter Added

A new configuration parameter has been added to allow a grace period during APL system shutdown so that B-tasks that detect APL shutdown have a chance to perform their own termination before being bounced by the B-task Scheduler during its termination.

The enhancement involves adding a parameter to the control vector stored in component 1of file `314159 btscstat`. The B-task Scheduler can be configured to delay a specified number of seconds between the time it detects that it should shut down and the time it actually starts to do so. This delay period is normally set to `¯1` (no delay), but can be reset using the *btsptch* function in workspace `314158` *opr*. For example:

```
13 btsptch 10
```

will specify a 10 second shutdown delay period. The left argument 13 is required; it specifies the position of the parameter in the control vector.

Changes to Operator Functions

crashfile (workspace 314158 *state/opr*) modified

The *crashfile* function may be used to shutdown any volume class, other than the master directory or, on a system with no master directory, the public class. Those classes must use the *crashfiles* function (since the file system cannot operate without them).

Users with files tied in the class being crashed are **not** bounced, nor are their files untied. However attempts to use these files return 'file library not available'. The tasks can untie the files, either by □*untie* or by signing off.

All outstanding file system program checks are cleared.

After the class has been 'crashed', it cannot be restarted, or dumped, until the crash recovery utility has been run against it.

See Chapter 11 of the *SHARP APL for MVS System Administrator's Guide* for details.

spstatus (workspace 314158 *opr/opfns*) modified

The output from the function has been improved. The new behavior is:

- When the wait code is 'I/O' the function displays the volser of the pack.
- When the wait code is 0 the function displays the wait code of '*pass*' (rather than declaring an unknown wait code).
- When the operation is 23, the function displays '*copy*' rather than '*(unused)*'.

See Chapter 11 of the *System Administrator's Guide* for details.

fadd (workspace 314158 *opr/opfns*) modified

The *fadd* function is less tolerant of negative library numbers. You now get a domain error for negative library number.

See Chapter 11 of the *System Administrator's Guide* for details.

finq added (workspace 314158 *opr/opfns*)

The *finq* function produces a tabular inquiry result for the volume classes in its left argument and the user numbers (and optional volume classes) in its right argument.

This new tool is useful in multi-volume class environments.

See Chapter 11 of the *System Administrator's Guide* for details.

lately added (workspace 314158 *opr/opfns*)

This new function returns a vector of task-ids that have been active during the most recent block of time. For example, *lately* 5 returns all tasks that have been active within the last 5 minutes.

See Chapter 11 of the *System Administrator's Guide* for details.

filestor added (workspace 314158 *state/opr*)

Syntax is: *filestor* '*fix/nofix,fix/nofix*'. It provides the functionality of the MVS modify sub-command FILESTOR.

See Chapter 11 of the *SHARP APL for MVS System Administrator's Guide* for details.

trapcnt added (workspace 314158 *state*)

As discussed in Chapter 2, a new □*trap* counter has been added to count how many times a task executes □*trap*. The function *trapcnt* is used to report on the count.

Syntax is: *r←interval trapcnt rows*

The result is an $n \times 2$ numeric matrix. The first column consists of task-ids. The second column consists of trap counts associated with those task-ids.

See Chapter 11 of the *SHARP APL for MVS System Administrator's Guide* for details.

trkhit (workspace 314158 *opr/opfns*) modified

The function *trkhit* now returns the track buffer hit rate based on the new EXTAB fields.

See Chapter 11 of the *System Administrator's Guide* for details.

Performance Package Functions - Documentation Added

Documentation for the APL functions *damax*, *maxspace*, *maxuser*, *setseg*, *tks*,*wscache*, and *wcmaxage*, first printed in the *V21 Internal and Operational Changes Guide*, has been placed in the *System Administrator's Guide*, in chapter 11, "The Operator Functions".

Changes to 1854339 *dump* Workspace

bsshist function added

A new function for returning file system histograms, *bsshist*, has been added to the *dump* workspace. The syntax is:

```
bsshist n
```

where *n* is the histogram number.

See Chapter 12 of the *SHARP APL for MVS System Maintenance Guide* for details.

wssi function modified

The system error analysis function *wssi* is used to analyze a failed workspace's *)si* stack. This function was displaying an incorrect value for the unused storage (*⎕wa*) in the workspace.

New Debugging Tool

A new workspace, 1854339 *fsdebug*, is distributed with V22. Most SHARP APL for MVS utilities and auxiliary processors support a debugging interface accessible from APL. The documentation for each utility describes the configuration information. The *fsdebg* workspace provides a convenient way for an APL task to interface with the debug subtask of a given utility or auxiliary processor.

See Chapter 12 of the *SHARP APL for MVS System Maintenance Guide* for details.

7 Environment Issues

IBM APAR OW22019

The following IBM PTF for APAR OW22019 is related to the way SHARP APL for MVS performs CHAPing operations. Customer operating systems should be updated to the PTF level indicated in the following table.

MVS level	*PTF#*
5.1	UW32552
5.2	UW32553
6.1(OS/390)	UW32554

The following is an excerpt from the IBM PTF notification:

```
*********************************************************************
* USERS AFFECTED: HBB5510 through JBB6602                           *
*********************************************************************
* PROBLEM DESCRIPTION: Performance degradation in APL due to        *
*                      IEAVSETS spinning holding the local          *
*                      lock repeatedly after APL CHAP requests      *
*********************************************************************
* RECOMMENDATION:                                                   *
*********************************************************************
APL processing invokes CHAP, to change the priority of another
Task in its address space.  While processing this request, CHAP
calls STATUS to signal other processors to stop processing in
the current address space.  This call to STATUS requres that
'local intersect' serialization be held, but CHAP is not holding
it (as of HBB5510).  Because this serialization is not held,
STATUS ends up spinning for an abnormally long amount of time
while waiting for all processors to stop running in the address
space.  This problem manifests as a performance degradation in
APL processing, because APL invokes CHAP repeatedly.
```

IBM APAR OW16355

The following is an excerpt from an IBM PTF notification:

```
*****************************************************************
* USERS AFFECTED: ALL ON HDZ11CO WHO SPECIFY CONTROL            *
*                 CHARACTERS FOR THEIR SPOOLED SYSIN            *
*                 OR SYSOUT DATA SETS                           *
*****************************************************************
* PROBLEM DESCRIPTION: In this case, JES3 sysout data sets did *
*                      not print correctly because the ASA      *
*                      control character was not processed.     *
*                      In general, the control character        *
*                      specification(A or M on the RECFM        *
*                      parameter) was being ignored for         *
*                      spooled sysin or sysout data sets.       *
*****************************************************************
* RECOMMENDATION:                                               *
*****************************************************************

IFG0196V was incorrectly clearing the control character flags
in the RECFM field of the spooled DCB. IFG0196V has been
modified to process correctly.
```

Configuring SRM for SHARP APL

In releases of MVS prior to MVS 5.1, it is possible to limit SRM's control of executing jobs and tasks by using the APGRNG parameter in an IEAIPSxx to define a limited range of dispatching priorities to be controlled by SRM. MVS tasks may then be run outside of SRM's control by assigning them explicit priorities outside the configured APGRNG.

SHARP APL is able to exploit this feature, using its own logic to change its dispatching priority as required; this mechanism is controlled by the APL configuration parameter PRIORITY and can be dynamically reset by the APL operator functions *aschap* and *priority*. For this mechanism to perform properly, the APL started task has to be run with a DPRTY that falls outside the APGRNG.

One of the changes introduced with MVS 5.1 is the elimination of the APGRNG; SRM controls all activity. This means that for SHARP APL systems at or beyond MVS 5.1, it is no longer advisable for APL to use its own mechanism to control its dispatching priority as doing so will contend against SRM.

To disable APL's mechanism, you must include the following line in one of the APL startup functions *startapl* or *startfsub*:

```
→aschap off ⍝apl chapping not possible at mvs 5.1 or beyond
```

As distributed, the *startapl* function includes a call to *aschap*.

Customers who wish to continue to control the CPU resources used by APL in the traditional way can do so by creating a time-sliced IPS group for APL. The SRM can be tuned to exert essentially the same control on APL's dispatching priority as APL itself used to maintain.

This is accomplished by creating an IEAIPSxx member of SYS1.PARMLIB that creates a time-sliced performance group with the desired characteristics, and by running the APL task within this performance group.

The following excerpts from a sample IEAIPSxx member illustrate this mechanism:

```
/*                                                                  */
/*   define time-slicing to give APL 90% at high, 10% at low        */
/*                                                                  */
/*                                                                  */
tsptrn=(5,5,5,5,5,5,5,5,5,*)          /* 90% APL, 10% not          */
/*                                                                  */
/* domain definitions                                               */
/*                                                                  */
/*                                                                  */
/*                                                                  */
dmn=4,dsrv=(1,999999999)          /* APL auxiliary processors       */
dmn=5,dsrv=(1,999999999)          /* APL itself.                    */
/*                                                                  */
/* priority scheme                                                  */
/*                                                                  */
/*   (6,1+10)  -  APL AP's (often overriden by dprty=)              */
/*   (6,0+10)  -  pgn 5 (APL) during time-slice intervals (hi)      */
/*                                                                  */
/*                                                                  */
/*                                                                  */
/*   (1,10)       APL (pgn 5) - base (low) priority                 */
/*                                                                  */
/*                                                                  */
/*                                                                  */
/*                                                                  */
/*   APL auxiliary processors, and APL itself                       */
/*                                                                  */
pgn=4,(dmn=4,dp=f61)                      /* APL ap's               */
pgn=5,(dmn=5,dp=f1,tsdp=f60,tsgrp=5)      /* APL task itself        */
```

Domain 5/performance group 5 is defined to run at a fixed priority of x'1a' (dp=f1) except when its priority is raised to x'6a' (tsdp=f60) by virtue of its membership in time-slicing group 5 (tsgrp=5). The time-slice pattern (tsptrn) is a vector showing, at each stage in a succession of SRM internal time intervals, which time-slice group is in its special state. In this case, being in the special state means running APL at its higher priority level, and the fact that there are 9 intervals in this state, and then a default interval ("*"), results in APL being at high priority 90% of the time.

This approach will need to be merged with whatever other definitions are in place in an IPS; these fragments are meant only to illustrate the pieces of the time-slicing mechanism.

8
APLMON

Overview

The changes made to the APLMON system for V22 are outlined below.

NOTE: The recommended minimum workspace size for SHARP APL is 450K. In APLMON, this minimum can be used with a minimal configuration of reports. If you experience a `wsfull`, reduce the number of reports and consider increasing workspace size. The optimal workspace size depends on the number of reports you want, but a suggested size is 800K.

New and Changed Reports

Three APLMON reports have been improved, and a new report has been added. A brief summary of each of the reports follows:

New Report: `bsstb` - the Track Buffer Report

The `bsstb` report provides information on track buffering for the file system. The report supports both total and rate (`bsstb.r`) reporting. The display of `bsstb` is limited to the first 20 volume classes.

Changes to the `bss` Report

The `bss` report has been expanded to show more information from the new file system histograms and to show track buffer information.

Changes to the `wscache` Report

The `wscache` report has been enhanced to provide additional information. Three new lines displaying "percent free", "forced swapouts", and "wscache address" have been added.

Changes to the *except* Report

A new class of exceptions, called the APL User Profile class, has been added to the *except* report. This class contains the member "trap count", which is used to identify tasks in a □*trap* loop.

For more detailed coverage of each of the reports, see the *APLMON User's Guide 3.2*.